View Reviews

Paper ID
701

Paper Title
TreeTracker Join: Killing Two Birds With One Stone

Track Name
Research -> September 2023

Reviewer #3

Questions

1. Overall Rating
Weak Reject

2. Relevant for PVLDB
Yes

3. Are there specific revisions that could raise your overall rating?
No

5. Paper Summary. In one solid paragraph, describe what is being proposed and in what context, and briefly justify your overall recommendation.
This paper studies how to smartly implement the Yannakakis algorithm for acyclic joins by integrating the join and semi-join into a single physical operator. It also shows that the theoretical optimality of Yannakakis is preserved. Moreover, it runs experiments to demonstrate its practical performance.

6. Three (or more) strong points about the paper. Please be precise and explicit; clearly explain the value and nature of the contribution.
S1. The problem is a very interesting and well motivated problem.
S2. It involves some interesting ideas about how to merge the bottom-up semi-join phase along the join tree and then a top-down retrieve join phase.
S3. This paper also shows strong theoretical guarantees on the method.

7. Three (or more) weak points about the paper. Please clearly indicate whether the paper has any mistakes, missing related work, or results that cannot be considered a contribution; write it so that the authors can understand what is seen as negative.
W1. The whole presentation requires significant improvement, which in some sense prevents readers from verifying the correctness of details, and appreciating the technical contribution. Please see 13.

W2. There are many works on this topic, such as semi-join methods and filter methods. But in the experimental study, this paper also compares their method with the baseline which uses hash join. It is unclear how the method proposed compares with other existing methods. Please also compare with the following work:
https://browse.arxiv.org/pdf/2307.15255.pdf

W3. Section 2 is to motivate the TTJ operator, but there is no other improvement we can get from binary join, which may not be a good example to . Computing the nest-loop already takes O(n^2) time, and the semi-join can be computed at the same time for free. But it is unclear how this will bring any benefit to multiway acyclic joins.

In algorithm 2.1, it is unclear to me about the ng (short for no-good list). As the input relation T does not contain duplicate tuples, every tuple t in T is distinct. Then, the if-condition at line 4 is always not satisfied, since no same tuple has been added to ng before. Moreover, in the main text, it says that “If t is known to not join with any s in S, the inner loop is skipped”. It does not make sense for this example, since all tuples t are distinct. How to know whether t can be joined with any s in S in advance for the binary join?

W4. W4. It claims that their reset-and-remove strategy can lead to an algorithm with O(n+r) data complexity. But this seems just a simple upward semi-join phase + join on the fly. The 3-way join example T(x), S(x,y,z) B(z) and 4-way join example T(x), S(x,y,z), B(z), R(y,z) are just two special, both of which just degenerate to semi-join between S and remaining relations. In these degenerated cases, the number of dangling intermediate join results is always bounded by |S|, and once an input tuple from S passes the filter, it can be outputted as the final join result. They are not general enough to illustrate.

8. Novelty. Please give a high novelty rating to papers on new topics, opening new fields, or proposing truly new ideas; give medium ratings to "delta" papers and those on well-known topics but still with some valuable contribution. (Note: For SDS and EA&B papers, novelty does not need to be in the form of new algorithms or models. Instead, novelty for SDS can be new understanding of issues related to data science technologies in the real world. Novelty for EA&B can be new insights into the strengths and weaknesses of existing methods or new ways to evaluate existing methods.)
With some new ideas

9. Significance
Improvement over existing work

10. Technical Depth and Quality of Content
Syntactically complete but with limited contribution

11. Experiments. (Reminder: EA&B papers should have a higher bar for experiments.)
OK, but certain claims are not covered by the experiments

12. Presentation
Sub-standard: would require heavy rewrite

13. Detailed Evaluation (Contribution, Pros/Cons, Errors). Please number each point and provide as constructive feedback as possible.
The notion of join query, acyclicity and all other definitions related to the join tree are not clearly, formally and precisely defined. For example:

A join tree is a tree where the nodes represent relations => A join tree is a tree where there is a one-to-one correspondence between relations and nodes

What does sort mean? Intuitively, it is a function that sort tuples in a relation in some ordering. But here, it is used for extracting attributes from a relation, which is quite counterintuitive.

The data complexity assumes the size of a query is a constant, not just a fixed parameter.

In relational algebra, the antijoin is well defined. Why does it introduce some other notations to denote R - (R semi-join S).

The structure in Section 3 is very messy. I suggest separating the background for conjunctive queries (acyclicity) and complexity measurement as the problem definition, then go to different notions in join tree, the left-deep query plan and its connection with join tree. All implementations regarding iterators can be moved to experiment.

There are many orderings defined but some of which are very confusing and unnecessary. For example, the join ordering on the join tree can be simplified as any top-to-down and left-to-right traversal of the join tree. Also, “W.l.o.g., we assume relations are labeled top-down in the same fashion as join operators R1 through Rk” In Figure

2 (a), all relations are labeled top-down Rk through R1. It is unclear which is the correct ordering for relations.

What is the difference between Ju and Ju^*? I suppose one should be the projection of join results onto attributes that appear in any relations with index larger than u, and the other should be the result computed by the left-deep tree. From the current description, there seems to be no difference.

What does this mean? “For a tuple t of $R(a, b)$, we use an unnamed perspective (e.g., $R(1, 2)$) to represent t [3].” tuple t has value 1 in attribute a and 2 in attribute b?

How to parse the following sentence? “TTJ assumes for a given relation Ri in PQ, its parent Rj in TQ is to the left of Ri , i.e., $j > i$.” In Figure 2, for relation R2 in PQ and its parent in TQ is R3. What does it mean that R3 is to the left of R2?

It is unclear what the two inputs and three outputs for the TTJ join operator. Algorithm 5.1 is hard to parse. Couldyou use plain language to explain the input, output, and steps of the procedure? The usage of class, void, open, get Next(), next, open() are very messy. It is helpful for readers to focus on the algorithmic idea here and go into all these low-level details in the experiment section.

Reviewer #4

Questions

1. Overall Rating
Weak Reject

2. Relevant for PVLDB
Yes

3. Are there specific revisions that could raise your overall rating?
Yes

5. Paper Summary. In one solid paragraph, describe what is being proposed and in what context, and briefly justify your overall recommendation.

The paper presents addresses the efficient execution of conjunctive queries using a variant of the hash join. The idea is to propagate
information about non-matching rows immediately when this is detected. The paper carefully proofs the correctness of the algorithm and
its optimility w.r.t. to the data complexity. The experiments show the potential improvements compared to using hash joins. However,
the paper does not compare against other related methods, e.g. using Bloom filters or simijoin reduction. Being limited to conjunctive queries
the practival relevance of the work remains unclear.

6. Three (or more) strong points about the paper. Please be precise and explicit; clearly explain the value and nature of the contribution.

S1: The correctness and optimality of the algorithm is analyzed at great detail which also helps to explain the algorithm of the TreeTrackerJoin.
S2: The experimental results show the potential of this operator in improving query performance.

7. Three (or more) weak points about the paper. Please clearly indicate whether the paper has any mistakes, missing related work, or results that cannot be considered a contribution; write it so that the

authors can understand what is seen as negative.

W1: The paper focuses on conjunctive queries without discussing how the ideas could be applied to a more comprehensive set of relational operators, e.g. outerjoins, group-by or sorting. This limits the practical relevance of the paper.
W2: Modern query engines use vectorized query processing to exploit the capabilties of modern hardware while the algoritm seems limited to a row-level iterator model of evaluation.
It would be interesting to understand how this limitation would affect the performance analysis of the paper.
W3: I would appreciate an analysis if and how communicating the "non-good" tuples contributes to the performance numbers, i.e. can we quantify the performance improvements relative to the detected non-matching rows?
W4: I wonder to what extent the TreeTrackerJoin could contribute to robust query processing, i.e. would sub-optimial plans still have close to optimal performance?
W5: Related to W4, I would appreciate an experimental analysis to other related efforts (mentioned in the paper), e.g. use of bloom filters, systems using the Yannakakis algoithms or worst-case optimal joins.

8. Novelty. Please give a high novelty rating to papers on new topics, opening new fields, or proposing truly new ideas; give medium ratings to "delta" papers and those on well-known topics but still with some valuable contribution. (Note: For SDS and EA&B papers, novelty does not need to be in the form of new algorithms or models. Instead, novelty for SDS can be new understanding of issues related to data science technologies in the real world. Novelty for EA&B can be new insights into the strengths and weaknesses of existing methods or new ways to evaluate existing methods.)
With some new ideas

9. Significance
Improvement over existing work

10. Technical Depth and Quality of Content
Solid work

11. Experiments. (Reminder: EA&B papers should have a higher bar for experiments.)
OK, but certain claims are not covered by the experiments

12. Presentation
Reasonable: improvements needed

13. Detailed Evaluation (Contribution, Pros/Cons, Errors). Please number each point and provide as constructive feedback as possible.

D1: Keeping track of the "non-good" tuples can add significant memory overhead - in the worst case most rows of a table are tracked. An experimental analysis of this aspect would improve the paper.
D2: In section 2, it would be good to mention the names of the operators for semijoin and antijoin when their symbols are used first.
D3: The paper would benefit from careful proof-reading of the language. Especially articles are missing in various cases. Some examples are:
- page 3: an article is missing in "TTJ modifies in-memory hash table"
- page 3: an article is missing in "the algorithm has to read input relations"
- an article is missing in "while deciding [the] oder for TTJ ... [18] with [the] TTK cost model"
etc.
- page 4: please fix the typo: "pseudocode" should be "pseudo code"
- page 4: the subject is missing in "when refer to method generically"
etc.

Reviewer #5

Questions

1. Overall Rating
Reject

2. Relevant for PVLDB
Yes

3. Are there specific revisions that could raise your overall rating?
No

5. Paper Summary. In one solid paragraph, describe what is being proposed and in what context, and briefly justify your overall recommendation.
This paper addresses the classic problem of relational k-way joins and general conjunctive (SPJ) queries. The authors present the TreeTrackerJoin which builds on top of a hash-join approach. In principle the idea is to detect and remove dangling tuples during the execution of the join instead of before as previous work does.

The paper conducts an extensive analysis, both practical and theoretical, to investigate the setting of TTJ, e.g., under which query plan is optimal.

Despite addressing an important problem with clear practical impact, I cannot recommend this work for publication.

6. Three (or more) strong points about the paper. Please be precise and explicit; clearly explain the value and nature of the contribution.
(S1) Important problem

(S2) Extensive analysis, both theoretical and practical

7. Three (or more) weak points about the paper. Please clearly indicate whether the paper has any mistakes, missing related work, or results that cannot be considered a contribution; write it so that the authors can understand what is seen as negative.
(W1) Contribution and novelty

(W2) Presentation and writing need significant improvement

(W3) Lack related work

(W4) Experimental analysis

8. Novelty. Please give a high novelty rating to papers on new topics, opening new fields, or proposing truly new ideas; give medium ratings to "delta" papers and those on well-known topics but still with some valuable contribution. (Note: For SDS and EA&B papers, novelty does not need to be in the form of new algorithms or models. Instead, novelty for SDS can be new understanding of issues related to data science technologies in the real world. Novelty for EA&B can be new insights into the strengths and weaknesses of existing methods or new ways to evaluate existing methods.)
Novelty unclear

9. Significance
Improvement over existing work

10. Technical Depth and Quality of Content
Syntactically complete but with limited contribution

11. Experiments. (Reminder: EA&B papers should have a higher bar for experiments.)
Obscure, not really sure what is going on and what the experiments show

12. Presentation
Sub-standard: would require heavy rewrite

13. Detailed Evaluation (Contribution, Pros/Cons, Errors). Please number each point and provide as constructive feedback as possible.
(1) My first concern about this work is related to its contributions and novelty; to be honest degree of both is unclear.
Essentially, TTJ builds on top of a straightforward hash-join which is extended with a deleteTR function to remove the dangling rows during the join process. The authors claim that the latter is a novel aspect but there is no extensive discussion of the related work to further inform the reader on it.

(2) As mentioned above, the related work is not extensively discussed. In fact, there is no related work section in the paper and therefore it's hard for the reader to position the paper in the existing literature and identify the competitive methods.

(3) The presentation and the writing need significant improvement.
- The paper is packed with technical details. Essentially, the methodology is presented very technically; even the pseudocode resembles actual code.
- The paper is packed with theorems and lemmas (15 in total). I understand the importance of the theoretical analysis but would be possible to move some of these in an appendix or at least discuss them in a less formal way.
- The notation introduced in Section 3 is hard to follow; a table summarising it would have very helpful.

(4) The experimental analysis also needs significant improvement.
- First of all, the authors consider only hash join (HJ) as a competitor to their TTJ.
- Very few experiments were included in the analysis using only two benchmarks JOB and TPC-H. How about real data?
Also, there are parameters which affect the performance of each solution, e.g., the number of join operators in the query plan, their selectivity, the cardinality of the inputs etc. Synthetic dataset were these parameters are controlled would provide a better insight for the performance gains of the proposed TTJ.